CSS niTe LP32

2014.2.15

Sassの日常の運用

または私は如何にして心配するのを止めて
プリプロセッサを愛するようになったか

2014.02.15

CSS Nite LP, Disk 32: Sass

富田 梓

誰？

名前：　富田 梓

所属：　LINE株式会社

職種：　マークアップエンジニア

活動：　@a_t

　　LINE Engineer ブログ

共著：　Sass入門

アジェンダ

1. Sassの導入と変遷

2. ライブラリの使い方

Sassの導入と変遷

Sassの導入と変遷

第1期 - 検証

実用できるかどうかを検査

第2期 - 導入

全体で利用開始

第3期 - 更新

問題点のフィードバックとライブラリへの反映

環境

プロジェクト

自社案件。MEはMEチームからアサイン。

CSSの作業者

MEのチームメンバー、Dなどが触る場合も

作業環境

OS、エディタは各自自分にあったものを利用

CSSのコーディングガイドライン

第1期 - 検証

- 2011年6月頃から
- まずは大きめのプロジェクトではじめる
- 実際に導入するかどうかはその後に判断


sass
_common_utility
_common_setting
_common_module
_common_layout
_md_common
_md_main
_md_sub
contentA
contentB
ミックスイン集
設定ファイル
汎用モジュール
レイアウト
モジュール
上記をimport

問題点

- 見通しの悪さ
- ファイル名などのルール化不足
- ミックスイン集の更新問題
- extendの使い方

第2期 - 導入

- チーム全体で導入の開始
- テンプレート作成
  （複数のプロジェクトで運用するため）
- チーム内での情報共有
- Compassの導入
  （主にspriteの自動生成などの機能）


sass
core
_setting
設定ファイル
_utility
ミックスイン集
_utility-css3
CSS3類
_style-mixin-base
_style-mixin-layout
_style-mixin-module
モジュール類
_core
coreをimport
_extend
extend
contentA
contentB
上記をimport

LINE Engineers' Blog

NAVERでのSassの使い方

by kosei_uemura on 2011.12.21

NAVER UITのSass Mixins職人（他称）の上村です。

今日はLess & Sass Advent calendar 2011の21日目です。

3日目に書かれている通り、NAVERでは半年ほど前から実務でSassを使っています。
今回は弊社で使用しているSassのディレクトリ/ファイル構成やMixinsについて解説します。

基本ディレクトリ/ファイル構成

今のところcss/sassディレクトリは下記のような構成を基本としています。

```
project
  |~css/
  |    |-category-A_TRUNK.css
  |    `-category-B_TRUNK.css
  |~sass/
       |~core/
```

問題点

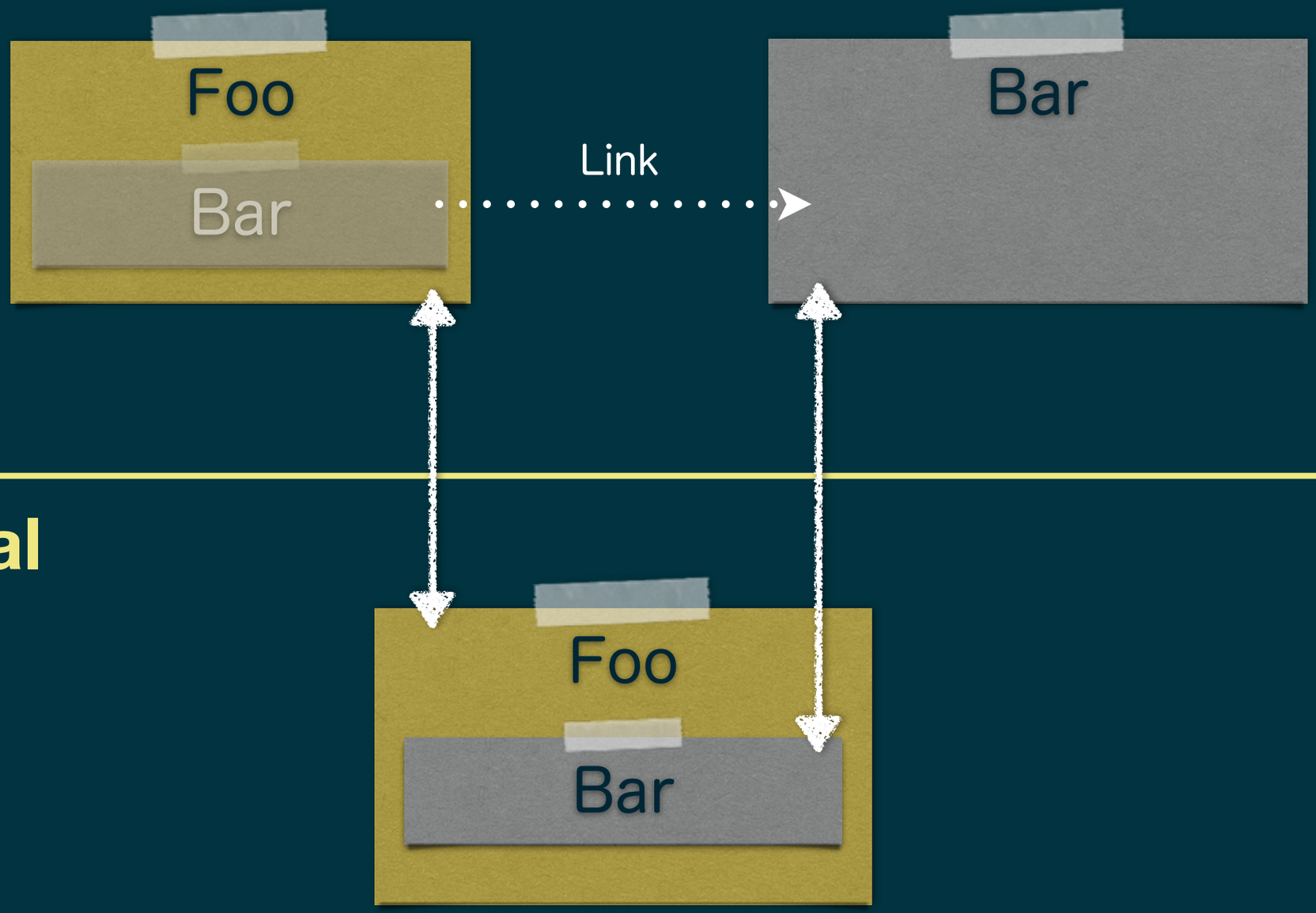
- 共通ファイルの更新問題
- 手動更新の限界
- CSS3のプロジェクトごとの切り分け

第3期 - 更新

- gitによるプロジェクト管理の開始
- ライブラリのsubmodule化
- 公開を視野に入れた更新

git submodule

Server



Local


lib
utility
_function
function類
_css3
CSS3類
_gradient
グラデーション
_reset
リセット
_general
その他
_import
上記をimport
_setting-css3
CSS3の設定
_setting
設定
_import
上記をimport


sass
lib
ライブラリ
parts
_extend
extend
_setting-lib
ライブラリ設定
_setting
設定
style-mixin
_base
_layout
_module
モジュール類
_import-default
parts、libをimprt
contentA
上記をimport

所感

- 小規模のプロジェクトではかなり冗長
- ミックスインで指定できるプロパティなど
  あらたな学習コストが発生
- 作業効率は向上
- プロジェクトの途中参加、交代も容易に
- 古いプロジェクトの修正が来ても
  「なんだっけ…」とか、ならなくなった

自社ライブラリの利用方法

の前に

- ライブラリは単純なものから始まり、
  要望に応える形でオプションを増やしてきた
- 全員がすべての機能を使いこなせている
  わけではない
- 部分的なものから使い始めて覚えていく

今日紹介するのは

- CSS3系のミックスイン
- グラデーション
- CSS Sprite

CSS3系の出力 - 初級

- 基本はプロパティと値の指定だけ
- どのベンダープレフィックスを出力するかは
  ライブラリ側で制御している

ライブラリ接頭辞

値

```sass
@include nj-column-count(3);
```

Sass

プロパティ名

```css
-webkit-column-count: 3;
-moz-column-count: 3;
-o-column-count: 3;
column-count: 3;
```

CSS

```sass
@include nj-box-sizing(border-box);
```

Sass

```css
-moz-box-sizing: border-box;
box-sizing: border-box;
```

CSS

```sass
@include nj-background-size(10px 20px);
```

Sass

```css
background-size: 10px 20px;
```

CSS

CSS3系の出力 - 上級

ベンダープレフィクスを制御したい場合

→ライブラリの設定を
プロジェクト側で上書きする

ライブラリでは次のように定義されている

sass/lib/_setting-css3.scss

```
// @:Multi-column Layout
//-----------------------------------------------------
$nj-prefixes-column-count        : webkit moz o !default;
:
```

```sass
@include nj-column-count(3);
```

Sass

```css
-webkit-column-count: 3;
-moz-column-count: 3;
-o-column-count: 3;
column-count: 3;
```

CSS

この設定をプロジェクト側のファイルで上書き

sass/parts/_setting_lib.scss

```
// @:CSS3
//-------------------------------------------------
$nj-prefixes-column-count          : webkit;
:
```

```sass
@include nj-column-count(3);
```

Sass

```css
-webkit-column-count: 3;
column-count: 3;
```

CSS

グラデーションの出力 - 初級

- 以下をまとめて出力
  - IE9向けSVG
  - webkit独自構文
    - ベンダープレフィックス付き
    - ベンダープレフィックス無し
- 方向は最も新しい形式（to right）に準拠

```sass
@include nj-linear-gradient(
to left, yellow, #33ccff);
```

Sass

```css
background-image: url(SVG…);
background-image: -webkit-gradient( linear,
100% 0, 0 0, from(yellow), to(#33ccff));
background-image: -webkit-linear-gradient(
right, yellow, #33ccff);
background-image: linear-gradient(
to left, yellow, #33ccff);
```

CSS

```sass
@include nj-linear-gradient(
to left, yellow, #33ccff);
```

Sass

```css
background-image: url(SVG…);
background-image: -webkit-gradient( linear,
100% 0, 0 0, from(yellow), to(#33ccff));
background-image: -webkit-linear-gradient(
right, yellow, #33ccff);
background-image: linear-gradient(
to left, yellow, #33ccff);
```

CSS

グラデーションの出力 - 上級

- IEのfilterを出力したい場合
- 角度(deg)指定で出力したい場合
- 構文を個別に出力したい場合

IEのfilterを出力したい場合

ライブラリの初期設定ではIE6,7とfilterがfalse

sass/lib/_setting.scss

```
// @:Support
//---------------------------------------------------
$nj-support-ie6    : false !default;
$nj-support-ie7    : false !default;
$nj-support-ie8    : true  !default;
:
// @:IE
//---------------------------------------------------
$nj-use-ie-filter    : false !default;
:
```

プロジェクト側のライブラリ設定ファイルでtrue

sass/parts/_setting_lib.scss

```
$nj-support-ie6     : true;
$nj-support-ie7     : true;
$nj-use-ie-filter   : true;
```

```sass
@include nj-linear-gradient(
to left, 000, #fff);
```

Sass

```css
:
background-image: linear-gradient(
to left, black, white);
*filter: progid:DXImageTransform.Microsoft.gradient(
gradientType=1,startColorStr='#FFFFFFFF',
endColorStr='#FF000000');
-ms-filter: "progid:DXImageTransform.Microsoft.gradient(
gradientType=1,startColorStr='#FFFFFFFF',
endColorStr='#FF000000')";
```

CSS

途中色は非対応のため、開始色と終了色のみ

Sass

```
@include nj-linear-gradient(
#232323, #f0f0f0 50%, #969696);
```

CSS

```
background-image: linear-gradient(
    #232323,
    #f0f0f0 50%,
    #969696);
*filter: progid …… gradient(
    gradientType=0,
    startColorStr='#FF232323',
    endColorStr='#FF969696');
```

角度(deg)指定で出力したい場合

方向の引数をdegで指定する

Sass

```
@include nj-linear-gradient(
135deg, blue, red);
```

CSS

```
background-image: url(SVG…);
background-image: -webkit-gradient(
linear, 0 0, 100% 100%, from(blue), to(red));
background-image: -webkit-linear-gradient(
315deg, blue, red);
background-image: linear-gradient(
135deg, blue, red);
```

対応していない角度の構文は出力しない

```sass
@include nj-linear-gradient(
83deg, blue, red);
```

Sass

```css
background-image: -webkit-linear-gradient(
7deg, blue, red);
background-image: linear-gradient(
83deg, blue, red);
```

CSS

構文を個別に出力したい場合

```sass
@include nj-legacy-linear-gradient(…)
```

Sass

```css
background-image: -webkit-linear-gradient(…);
```

CSS

```sass
@include nj-svg-linear-gradient(to right,
```

Sass

```css
background-image: url(SVG…);
```

CSS

Spriteの出力 - 初級

- Spriteの出力には、次の3つの指定が必要
  - 対象の画像ディレクトリ
  - Sprite画像のパス
    (background-image)
  - 個別画像のサイズとポジション
    (width、height、background-position)

sass/parts/_setting-compass-sprite.scss

```
$spr-map-sp01: nj-sprite-map(
    "/img/sprite/sp01",
    $layout: horizontal,
    $spacing: 0px
);
```

Sass

```
%spr-base-sp01 {
    @include nj-compass-spr-base($spr-map-sp01);
}
hoge{
    @extend %spr-base-sp01;
    @include nj-compass-spr($spr-map-sp01, "hoge");
}
```

sass/parts/_setting-compass-sprite.scss

```
$spr-map-sp01: nj-sprite-map(
     "/img/sprite/sp01",
     $layout: horizontal,
     $spacing: 0px
);
```

1.対象ディレクトリ

Sass

```
%spr-base-sp01 {
     @include nj-compass-spr-base($spr-map-sp01);
}
hoge{
     @extend %spr-base-sp01;
     @include nj-compass-spr($spr-map-sp01, "hoge");
}
```

2. background-image

3. width、height、background-position

sass/parts/_setting-compass-sprite.scss

```
$spr-map-sp01: nj-sprite-map(
    "/img/sprite/sp01",
    $layout: horizontal,
    $spacing: 0px
);
```

```
%spr-base-sp01 {
    @include nj-compass-spr-base($spr-map-sp01);
}
hoge{
    @extend %spr-base-sp01;
    @include nj-compass-spr($spr-map-sp01, "hoge");
}
```

Sass

CSS

```
hoge{
  background-image: url("/img/sprite/sp01.png");
}
hoge {
  width: 24px;
  height: 24px;
  background-position: -38px 0;
}
```

spriteファイルを追加する場合

Sass

```
%spr-base-sp01 {
    @include nj-compass-spr-base($spr-map-sp01);
}
hoge{
    @extend %spr-base-sp01;
    @include nj-compass-spr($spr-map-sp01, "hoge");
}
fuga{
    @extend %spr-base-sp01;
    @include nj-compass-spr($spr-map-sp01, "fuga");
}
```

この部分だけ追加する

CSS

```css
hoge, fuga {
  background-image: url("/img/sprite/sp01.png");
}
hoge {
  width: 24px;
  height: 24px;
  background-position: -38px 0;
}
fuga {
  width: 32px;
  height: 25px;
  background-position: -62px -32px;
}
```

画像のサイズを出力したくない場合

Sass

```
%spr-base-sp01 {
    @include nj-compass-spr-base($spr-map-sp01);
}
hoge{
    @extend %spr-base-sp01;
    @include nj-compass-spr(
        $spr-map-sp01,
        “hoge”,
        $_width: null,
        $_height: null);
}
```

画像のサイズを出力したくない場合

CSS

```
hoge{
  background-image: url(“/img/sprite/sp01.png”);
}
hoge {
  background-position: -38px 0;
}
```

Spriteの出力 - 上級

- Retinaに対応したSpriteを出力したい場合
- 画像のキャッシュを防ぐクエリーを付けたい
- Spriteの多用でコンパイルが重くなった場合

Retinaに対応したSpriteを出力したい場合

Sass

```
$spr-map-sp01: nj-sprite-map(
    "/img/sprite/sp01",
    $layout: horizontal,
    $spacing: 10px
);
%spr-base-sp01 {
    @include nj-compass-retina-spr-base(
    $spr-map-sp01);
}
hoge{
    @extend %spr-base-sp01;
    @include nj-compass-retina-spr(
    $spr-map-sp01, "hoge");
}
```

```css
hoge{
  background-image: url(“/img/sprite/sp01.png”);
  background-size: 67px auto;
}
hoge {
  width: 12px;
  height: 12px;
  background-position: -19px 0;
}
```

画像のキャッシュを防ぐクエリーを付けたい

Sass

```
%spr-base-sp01 {
    @include nj-compass-spr-base(
        $spr-map-sp01,
        $_parameter:'?20140215_01'
    );
}
hoge{
    @extend %spr-base-sp01;
    @include nj-compass-spr($spr-map-sp01, "hoge");
}
```

```css
hoge{
  background-image: url(
    “/img/sprite/sp01.png?20140215_01”
  );
}
```

```css
hoge {
  width: 24px;
  height: 24px;
  background-position: -38px 0;
}
```

Spriteの多用で
コンパイルが重くなった場合

コンパイル時のSprite処理を停止する

sass/parts/_setting-compass-sprite.scss

```
// # Compass Spriteのオン／オフ

$nj-use-compass-sprite: false;
```

sprite画像（結合した画像）ではなく、
結合前の個別画像が出力されるので、
表示上は崩れない

Sassファイルの変更は不要

Sass

```
%spr-base-common {
      @include nj-compass-spr-base($spr-map-common);
}
hoge{
      @extend %spr-base-common;
      @include nj-compass-spr($spr-map-common, "hoge");
}
fuga{
      @extend %spr-base-common;
      @include nj-compass-spr($spr-map-common, "fuga");
}
```

CSS

```
hoge, fuga {
  background-image: url("/img/sprite/sp01.png");
}
hoge {
  width: 24px;
  height: 24px;
  background-position: -38px 0;
}
fuga {
  width: 32px;
  height: 25px;
  background-position: -62px -32px;
}
```

CSS

```
hoge, fuga {
 /* Compass Sprite: オフ状態 */
}
hoge {
  width: 24px;
  height: 24px;
  background-image:url("/img/sprite/sp01/hoge.png");
}
fuga {
  width: 32px;
  height: 25px;
  background-image:url("/img/sprite/sp01/fuga.png");
}
```

まとめ

- 最大の魅力はニーズに合わせて
  柔軟にカスタマイズできること

- 自分/自社のやりかたに合った
  “秘伝のタレ”を目指しましょう

いつもの

LINE株式会社では一緒に働いてくれる
フロントエンドエンジニアを絶賛募集中です。
渋谷ヒカリエで僕と握手！

https://linecorp.com/career

ありがとうございました。